Вінницький технічний фаховий коледж

Циклова комісія комп’ютерної інженерії

**КУРСОВИЙ ПРОЄКТ**

з програмування

(назва дисципліни)

на тему: *Програма обліку інформації про країни та регіони*

Студента 4 курсу 4КІ1 групи

галузі знань 12 Інформаційні технології

спеціальності 123 Комп’ютерна інженерія

Опанасюка І. О.

(прізвище та ініціали)

Керівник Андрій СТЕЦЬ

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_\_\_\_\_Оцінка: ECTS

Члени комісії

(підпис) (прізвище та ініціали)

(підпис) (прізвище та ініціали)

(підпис) (прізвище та ініціали)

ВІННИЦЯ 2023

ЗМІСТ

[Анотація 4](#_Toc146624802)

[Вступ 5](#_Toc146624803)

[1 Аналіз задачі та предметної області 6](#_Toc146624804)

[1.1 Аналіз предметної області 6](#_Toc146624805)

[1.2 Аналіз та формування тестових даних 9](#_Toc146624806)

[1.3 Аналіз архітектурного рішення 12](#_Toc146624807)

[1.4 …. 12](#_Toc146624808)

[2 Реалізація (теоретичної індивідуальної теми) в проєкті 13](#_Toc146624809)

[2.1 …. 13](#_Toc146624810)

[2.2 …. 13](#_Toc146624811)

[3 Проєктна частина 14](#_Toc146624812)

[3.1 Реалізація класів сутностей ПО та контексту даних 14](#_Toc146624813)

[3.2 Реалізація команд перегляду даних 21](#_Toc146624814)

[3.3 Реалізація засобів файлового введення-виведення 23](#_Toc146624815)

[3.4 Реалізація меню та циклу виконання програми 24](#_Toc146624816)

[3.5 …. 24](#_Toc146624817)

[Висновки 25](#_Toc146624818)

[Список використаних джерел 26](#_Toc146624819)

[Додатки 28](#_Toc146624820)

[Додаток А 29](#_Toc146624821)

[Додаток Б 30](#_Toc146624822)

[Додаток В 31](#_Toc146624823)

[Додаток Г 34](#_Toc146624824)

[Додаток Д 36](#_Toc146624825)

[Додаток Е 37](#_Toc146624826)

[Додаток Ж 38](#_Toc146624827)

# Анотація

Завданням даного курсового проєкту є розробка програми-менеджера власних фінансів.

Дане програмне рішення, що було реалізовано засобами MS Visual Studio на платформі ASP.NET і Blazor, є веб-додатком для управління особистими фінансами. Використовуючи такі технології, як Entity Framework, MS SQL Server, Blazor Server та MudBlazor, проєкт надає користувачу зручний інтерфейс для ведення фінансового обліку та генерування звітів.

У пояснювальній записці до проєкту описано структуру і процес створення проєкту, а також детально розглянуто можливості бібліотеки MudBlazor та особливості її використання в контексті реалізації проєкту.

# Вступ

Завданням даного курсового проєкту є розробка програми-менеджера власних фінансів.

Створена програма забезпечує можливість редагування, додавання, видалення та читання інформації з бази даних, а також отримання звітів за певний період часу.

Реалізовано засоби редагування даних із застосуванням веб-інтерфейсу, що забезпечує можливість перегляду і редагування інформації про об’єкти предметної області з використанням mudblazor-форм редагування даних в БД.

Програмне рішення було створене у середовищі MS Visual Studio з використанням мови програмування C# та платформи Blazor Server та ASP.NET Web-API.

У процесі виконання завдання були реалізовані 5 проектів:

1. Проект SelfFinanceAPI.Core, який містить логіку взаємодії з базою даних, включаючи класи репозиторіїв та сервісів.
2. Проект SelfFinanceAPI, який описує API та його контролери.
3. Проект SelfFinanceAPI.Tests, де розроблено модульні тести для функцій контролерів, а також класів репозиторіїв та сервісів з проектів SelfFinanceAPI та SelfFinanceAPI.Core.
4. Проект SelfFinanceServer, що містить веб-інтерфейс користувача для перегляду та редагування даних.
5. Допоміжний проект SelfFinanceCommon, де описані data transfer objects (DTOs), які використовуються як для веб-інтерфейсу, так і для API.

Реалізовано засоби для роботи із базою даних MS SQL. Для виконання З метою забезпечення ефективного взаємодії з базою даних MS SQL використовується бібліотека Entity Framework, а міграції дозволяють автоматично створювати та заповнювати базу даних при запуску програми, якщо база даних не існує.

Веб-інтерфейс реалізовано за допомогою веб-фреймворка Blazor у взаємодії із фреймворком компонентів UI MudBlazor.

Blazor – це UI-фреймворк для створення інтерактивних додатків, які можуть працювати як на стороні сервера, так і на стороні клієнта, на платформі .NET.

MudBlazor – це фреймворк компонентів Material Design для Blazor з акцентом на простоту використання та чітку структуру. Він ідеально підходить для .NET розробників, щоб створювати веб-додатки без необхідності поглиблюватися в CSS та Javascript.

Загалом, за допомогою Blazor уникнуто складних скриптів JavaScript, забезпечуючи легку інтеграцію та розробку інтерфейсу користувача на C#, а MudBlazor дозволяє не поглиблюватись в нюанси написання розмітки HTML та CSS-стилів, за рахунок чого значно полегшується розробка інтерфейсу користувача.

# Аналіз задачі та предметної області

## Аналіз предметної області

Досліджуваною предметною областю є ПО «Повсякденні фінансові операції». Дана предметна область охоплює інформацію про сутності:

* Тип витрат – тип фінансової операції, який може бути як прибутковий, так і збитковий;
* Фінансова операція – безпосередньо сама фінансова операція, яку здійснює користувач і реєструє в додатку (має свій об’єкт типу, і додадкові параметри, які її відрізняють, як час і дата, розмір і т.п.);
* Звіт – об’єкт, який не зберігається в базі даних, але утворюється за відповідним запитом, містить інформацію про фінансові операції за певний проміжок часу.

Сутності ПО та атрибути сутностей представлені класами та їх членами (таблиця 1.1).

Назви класів сутностей та характеристики полів, що призначені для збереження значень атрибутів сутностей, представлені в таблицях 1.2 – 1.4.

Таблиця 1.1 – Сутності ПО та їх атрибути

|  |  |
| --- | --- |
| Сутність | Атрибути |
| Тип витрат | Назва, Чи є прибутком |
| Фінансова операція | Тип, Сума, Дата і час, Примітка |
| Звіт | Початкова дата звіту, Кінцева дата звіту, Загальний прибуток, Загальні витрати, Список операцій |

Таблиця 1.2 – Опис полів класу ExpenseType (сутність Тип витрат)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Атрибут | Назва поля | Тип даних | Є обов’яз­ковим |  |
| Ідентифікатор | Id | int | так |  |
| Назва | Name | string | так | Рядок від 1-го символа |
| Чи є прибутком | IsIncome | boolean | так | За замовчуванням false |

Таблиця 1.3 – Опис полів класу FinancialOperation, що представляє сутність Фінансова операція

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Атрибут | Назва поля | Тип даних | Є обов’яз­ковим |  |
| Ідентифікатор | Id | int | Так |  |
| Ідентифікатор типу | TypeId | int | Так |  |
| Сума | Amount | decimal | Так | Невід’ємне |
| Дата і час | DateTime | DateTime | Так | Не більше поточної |
| Опис | Description | string | Ні |  |

Таблиця 1.4 – Опис полів класу Report, що представляє сутність Звіт

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Назва поля | Тип даних | Є обов’яз­ковим |
| 1 | 2 | 3 | 4 |
| Початкова дата | startDate | Country | Так |

Продовження таблиці 1.4

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| Кінцева дата | endDate | DateTime | Так |
| Загальний прибуток | totalIncome | String | Так |
| Загальні витрати | totalExpenses | decimal | Так |
| Список операцій | operations | List<FinancialOperation> | Так |

## Аналіз та формування тестових даних

Формування набору тестових даних доцільно почати з відбору даних для сутності Чисельність населення, оскільки її дані допоможуть визначитись із обов’язковим переліком країн, дані про які потрібно включити у набір тестових даних.

Результуюча вибірка інформації про чисельність населення країн представлені у таблиці 1.5 [2-6].

Таблиця 1.5 – Вибрана інформація про чисельність населення країн (мінімальний набір)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Країна | Дата | Значення  (чоловік) | Є результ. перепису | Примітка |
| Україна | 01.01.2014 | 45426200 | ні |  |
| Україна | 01.02.2014 | 43057280 | ні | без урахування Криму |
| Боснія і Герцеговина | 15.10.2013 | 3531159 | так | остаточні дані перепису |
| Північна Корея | 01.10.2014 | 24213510 | так | попередні дані перепису |
| Україна | 01.01.2014 | 45426200 | ні |  |
| Україна | 01.02.2014 | 43057280 | ні | без урахування Криму |

На основі даних про назви країн з таблиці 1.5 можна сформувати мінімальний перелік країн, дані про які потрібно ввести в якості тестових.

Результуюча вибірка інформації про країни та географічні регіони представлені у таблицях 1.6 та 1.7 [7, 8].

Таблиця 1.6 – Вибрана інформація про країни (мінімальний набір)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Назва | Двохлітерний код | Доменне ім'я | Географічний регіон | Площа, км2 |
| Україна | UA | .ua | Східна Європа | 603628 |
| Боснія і Герцеговина | BA | .ba | Східна Європа | 51129 |
| Ватикан | VA |  | Південна Європа | 0,44 |
| Сполучені Штати Америки | US |  | Північна Америка |  |

Таблиця 1.7 – Вибрана інформація про країни (багаторядковий текст)

|  |  |  |
| --- | --- | --- |
| Країна | Атрибут | Значення |
| Україна | Опис | Украї́на – держава у Східній Європі та частково в Центральній Європі, у південно-західній частині Східноєвропейської рівнини. Межує з Білоруссю на півночі, Польщею, Словаччиною та Угорщиною – на заході, Румунією та Молдовою – на південному заході, Росією на сході і північному сході. На півдні і південному сході омивається Чорним і Азовським морями [9] |
| Україна | Примітка | 603549 км2 – загальна площа України з урахуванням Криму. Без Автономної Республіки Крим та Севастополя, які окуповані і фактично контролюються Російською Федерацією, але входження яких до її складу не визнається ООН, площа України становить 576604 км2. [10] |

На основі даних про назви країн з таблиці 1.6 можна сформувати мінімальний перелік географічних регіонів, дані про які потрібно ввести в якості тестових.

Результуюча вибірка представлена у таблицях 1.8 та 1.9.

Таблиця 1.8 – Вибрана інформація про географічні регіони (мінімальний набір)

|  |  |  |
| --- | --- | --- |
| Назва | Назва географічного регіону вищого рівня | Площа, км2 |
| Світ | (не існує) | 136162000 |
| Азія | Світ | 31915000 |
| Європа | Світ | 23049000 |
| Східна Європа | Європа |  |
| Південна Європа | Європа |  |
| Південна Америка | Америка | 17832000 |

Таблиця 1.9 – Вибрана інформація про географічні регіони (багаторядковий текст)

|  |  |  |
| --- | --- | --- |
| Географічний регіон | Атрибут | Значення |
| 1 | 2 | 3 |
| Світ | Опис | Світ – назва планети Земля з людської точки зору, як місце заселене людськими істотами [11]. |
| Азія | Опис | А́зія – найбільша за площею (біля 43,4 мільйонів км²) і за населенням частина світу, розташована головним чином у Східній півкулі. Разом з Європою утворює материк Євразію [12]. |

Продовження таблиці 1.9.

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| Європа | Опис | Євро́па (іноді Евро́па) – частина світу в Північній півкулі.  Європа омивається Північним Льодовитим океаном на півночі, Атлантичним океаном на заході та Середземним морем на півдні. На сході та південному сході від Азії Європу відділяють Уральські та Кавказькі гори, річка Урал (за іншою версією, річка Емба), Каспійське та Чорне моря [13]. |
| Європа | Примітка | Для визначення приналежності країн до географічних регіонів Європи використано поділ Європи, що прийнятий у ООН [14]. |

## Аналіз архітектурного рішення

….

## ….

….

# Реалізація (теоретичної індивідуальної теми) в проєкті

## ….

….

### …

## ….

….

# Проєктна частина

## Реалізація класів сутностей ПО та контексту даних

Для реалізації зв’язків між записами з даними про об’єкти сутностей у сховищі в кожному з класів сутностей існує наслідування класу Entity, що містить значення унікального числового ідентифікатора. Доступ до значення числового ідентифікатора забезпечує властивість Id типу int, яка реалізується компілятором автоматично.

У якості назви класу, що представляє країни використано ідентифікатор Country. Діаграма класу представлена на рисунку 3.1.

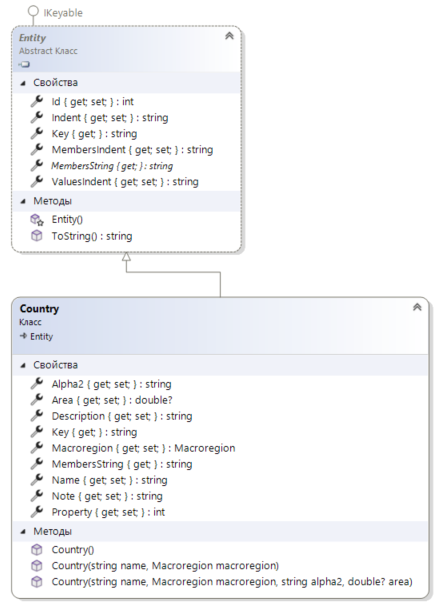


Рисунок 3.1 – Діаграма класу Country

Параметризовані конструктори класу (лістинг 3.1, 3.2) забезпечують можливість зручної ініціалізації полів об’єкта класу у момент його створення.

Лістинг 3.1 – Параметризований конструктор класу Country

public Country(string? name, Macroregion? macroregion,

string? alpha2, double? area) {

Name = name;

Macroregion = macroregion;

Alpha2 = alpha2;

Area = area;

}

public Country(string name, Macroregion macroregion)

: this(name, macroregion, null, null) { }

public Country() : this(null, null, null, null) { }

Лістинг 3.2 – Параметризований конструктор класу Macroregion

public Macroregion(string? name, Macroregion? parent, double? area) {

Name = name;

Parent = parent;

Area = area;

}

public Macroregion() : this(null, null, null) { }

public Macroregion(string name, Macroregion? parent)

: this(name, parent, null) { }

Перший параметр конструктора обов'язковий, а для інших параметрів встановлені значення за замовчуванням. Такий підхід дозволяє використовувати цей конструктор з різними наборами аргументів, навіть якщо аргументи надаються не в обов'язковому порядку. Використання іменованих аргументів дозволяє змінювати порядок їх передачі у списку аргументів конструктора.

Для ініціалізації частини полів можна також використати ініціалізатор об’єкта. Приклад використання конструктора для ініціалізації полів об’єктів у методі створення тестового набору даних класу даних TestingDataMethods показано у лістингу 3.3 та лістинг 3.4.

Лістинг 3.3 – Створення тестового набору об’єктів метод CreateTestingMacroregions

private static void CreateTestingMacroregions(ICollection<Macroregion> macroregions) {

macroregions.Add(new Macroregion("Світ", null, 130094010) {

Id = 1,

Description = "Світ — назва планети Земля з людської " +

"точки зору, як місце заселене людськими істотами.\n" +

"(https://uk.wikipedia.org/wiki/Світ)"

});

macroregions.Add(new Macroregion("Європа",

macroregions.FirstOrDefault(e => e.Name == "Світ"), 22134900) {

Id = 2,

Description = "Євро́па (іноді Евро́па) — частина світу " +

"в Північній півкулі.\nЄвропа омивається Північним " +

"Льодовитим океаном на півночі, Атлантичним океаном " +

"на заході та Середземним морем на півдні. На сході " +

"та південному сході від Азії Європу відділяють " +

"Уральські та Кавказькі гори, річка Урал (за іншою " +

"версією, річка Емба), Каспійське та Чорне моря.\n" +

"(https://uk.wikipedia.org/wiki/Європа)"

});

macroregions.Add(new Macroregion("Східна Європа",

macroregions.FirstOrDefault(e => e.Name == "Європа")) { Id = 3 });

macroregions.Add(new Macroregion("Південна Європа",

macroregions.FirstOrDefault(e => e.Name == "Європа")) { Id = 4 });

macroregions.Add(new Macroregion("Північна Америка", null, 18651660) { Id = 5 });

macroregions.Add(new Macroregion("Південна Америка", null, 17461112) { Id = 6 });

}

Лістинг 3.4 – Створення тестового набору об’єктів метод CreateTestingCountries

private static void CreateTestingCountries(IDataSet dataSet) {

dataSet.Countries.Add(new Country("Україна",

dataSet.Macroregions.First(e => e.Name == "Східна Європа"), "UA", 603500) {

Id = 1,

Note = "Площа України вказана у відповідності зі " +

"статистичними даними ООН станом на 2019-й рік " +

"(http://data.un.org/en/iso/ua.html).",

Description = "Украї́на — держава у Східній Європі та частково " +

"в Центральній Європі, у південно-західній частині " +

"Східноєвропейської рівнини.Межує з Білоруссю на півночі,"+

"Польщею, Словаччиною та Угорщиною — на заході, Румунією" +

"та Молдовою— на південному заході, Росією на сході і " +

"північному сході.На півдні і південному сході омивається"+

"Чорним і Азовським морями.\nСучасна Україна, обравши " +

"за свій герб знак княжої держави Володимира Великого, " +

"проводить свою державність від Русі київських князів " +

"династії Рюриковичів IX—XIII століть.\n" +

"(https://uk.wikipedia.org/wiki/Україна)"

});

dataSet.Countries.Add(new Country("Боснія і Герцеговина",

dataSet.Macroregions.First(e => e.Name == "Південна Європа")) {

Id = 2,

Area = 51129,

});

dataSet.Countries.Add(new Country("Ватикан",

dataSet.Macroregions.First(e => e.Name == "Південна Європа"), "VA", 0.44) {

Id = 3,

});

dataSet.Countries.Add(new Country("Сполучені Штати Америки",

dataSet.Macroregions.First(e => e.Name == "Північна Америка")) {

Id = 4,

Alpha2 = "US",

});

dataSet.Countries.Add(new Country("Білорусь",

dataSet.Macroregions.First(e => e.Name == "Східна Європа"), "BY", 207600) {

Id = 5,

});

dataSet.Countries.Add(new Country("Болгарія",

dataSet.Macroregions.First(e => e.Name == "Східна Європа"), "BG", 110910) {

Id = 6,

});

}

Метод розширення CreateTestingData для інтерфейсу IDataSet дозволяє додавати нові функціональність інтерфейсам без модифікації їхнього вихідного коду. У цьому випадку, метод CreateTestingData (лістинг 3.5) розширює інтерфейс IDataSet, додаючи можливість створення тестових даних для нього. Метод CreateTestingMacroregions використовується у програмі для збереження даних про об’єкти сутності ПО та виконання операцій з ними. Метод CreateTestingCountries працює по аналогії з методом CreateTestingMacroregions.

Лістинг 3.5 – Реалізація колекції об’єктів класу сутності Countries

public static void CreateTestingData(this IDataSet dataSet){

CreateTestingMacroregions(dataSet.Macroregions);

CreateTestingCountries(dataSet);

}

У класах також описано конструктор без параметрів, який використовується за замовчуванням і, зокрема, необхідний для роботи механізму серіалізації у форматі XML.

Для формування текстового представлення екземплярів класу, яке включає значення усіх полів об’єкта, у класі Country та класі Macroregion перевизначено загальнодоступний віртуальний метод MembersString (лістинги 3.6, 3.7). Метод повертає відформатований рядок у якому для значення кожного поля визначена ширина результуючого рядка і спосіб вирівнювання значення у ньому. Це забезпечує відображення даних колекції об’єктів сутності у вигляді таблиці, яка є зручною для перегляду і аналізу даних (рисунок 3.2).

Властивість Key (лістинг 3.8) визначає ключ або ідентифікатор об'єкту. Вона повертає значення Name, якщо Name не є null, інакше вона використовує значення базового ключа base.Key. Тобто ключем об'єкта в цьому випадку є ім'я країни, але якщо ім'я країни є null, то ключем стає значення, отримане від базового класу. За аналогією працює і для макрорегіону.

Лістинг 3.6 – Перевизначення методу MembersString класу Country

public override string MembersString =>

$"{MembersIndent}Назва: {Name}\n" +

$"{MembersIndent}Географічний регіон: {Macroregion?.Name}\n" +

$"{MembersIndent}Код alpha-2 ISO: {Alpha2}\n" +

$"{MembersIndent}Площа, км: {Area}\n" +

$"{MembersIndent}Примітка: {Note?.ToTextBlocks(ValuesIndent, "\n")}\n" +

$"{MembersIndent}Опис: {Description?.ToTextBlocks(ValuesIndent, "\n")}";

Лістинг 3.7 – Перевизначення методу MembersString класу Macroregion

public override string MembersString =>

$"{MembersIndent}Назва: {Name}\n" +

$"{MembersIndent}Макрорегіон: {Parent?.Name}\n" +

$"{MembersIndent}Площа, км: {Area}\n" +

$"{MembersIndent}Опис: {Description?.ToTextBlocks(ValuesIndent, "\n")}";

Лістинг 3.8 – Перевизначення методу Key

public override string Key => Name ?? base.Key;

Повний код реалізації класу Country наведений у додатку А.

Повний код реалізації класу Macroregion наведений у додатку Б.

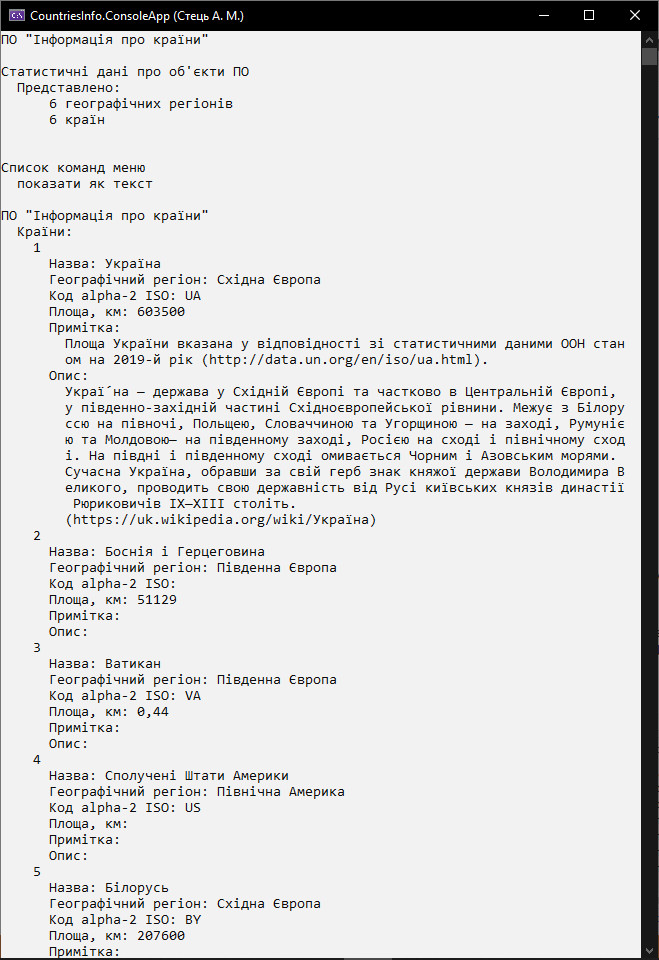


Рисунок 3.2 – Відображення даних списку об’єктів сутності ПО

## Реалізація команд перегляду даних

Для реалізації команд відображення даних створено окремий статичний клас FormattingMethods. Цей клас представляє собою розширення для форматування даних у консольному додатку, пов'язаному з інформацією про географічні регіони (макрорегіони) і країни.

Метод ToDataString (лістинг 3.9) призначений для форматування загальних даних про географічні регіони і країни. Він приймає набір даних через інтерфейс IReadOnlyDataSet. Метод створює рядок, що містить заголовок та дані про країни та географічні регіони. Залежно від наявності контексту відступів, дані можуть відступати або виводитися без відступів.

Лістинг 3.9 – Опис методу ToDataStrin

public static string ToDataString(this IReadOnlyDataSet dataSet,

string? header = null) {

header ??= "ПО \"Інформація про країни\"";

string? s = null;

if (IndentContextMediation.Enabled) {

s = $"{LineIndent.Current.Value}{header}\n";

LineIndent.Current.Increase();

s = string.Concat(s, dataSet.Countries.ToIndentedLineList("Країни"),

dataSet.Macroregions.ToIndentedLineList("Географічні регіони"));

LineIndent.Current.Decrease();

}

else {

s = string.Concat(header == null ? "" : $"{header}\n",

dataSet.Countries.ToLineList("Країни"), "\n",

dataSet.Macroregions.ToLineList("Географічні регіони"));

}

return s;

}

Також метод ToTable (рисунок 3.2) для класу Macroregion та класу Country форматує дані об'єктів Macroregion і Country у виді табличного представлення (лістинг 3.10) з різними стовпцями. Таблиця виводиться у форматі, де кожна колонка має фіксований розмір, щоб забезпечити візуальну чіткість .

Лістинг 3.10 – Фрагмент опису часткового класу FormattingMethods

public static string? ToTable(this IEnumerable<Country> objects,

string? header = null) {

StringBuilder sb = new StringBuilder();

sb.Append(header ?? "Країни і залежні території");

sb.AppendLine();

if (!objects.Any()) {

sb.AppendLine("\t(дані відсутні)");

return sb.ToString();

}

string format = " {0,5} {1,-29} {2,-20} {3,7} {4,12}\n";

sb.AppendFormat(format, "Id","Назва","Географічний регіон",

"Alpha-2", "Площа, км²");

sb.AppendFormat($" {new string('-', 77)}\n"); // Що робити зі значенням 77

foreach (var obj in objects) {

sb.AppendFormat(format, obj.Id, obj.Name, obj.Macroregion?.Name,

obj.Alpha2, obj.Area);

}

return sb.ToString();

}

Метод ToStatisticsString (лістинг 3.11) форматує статистичні дані про кількість географічних регіонів і країн у наборі даних.

Лістинг 3.11 – Опис методу ToStatisticsString

public static string ToStatisticsString (this IReadOnlyDataSet dataSet,string? header = null) {

return $"{header ?? "Статистичні дані про об'єкти ПО"}\n" +

$" Представлено:\n"

+ $"{dataSet.Macroregions.Count(),7} географічних регіонів\n"

+ $"{dataSet.Countries.Count(),7} країн\n";

}

## Реалізація засобів файлового введення-виведення

Для реалізації файлового введення-виведення у класі DataContext описано статичне загальнодоступне поле fileName, що визначає ім’я файлу з каталогу додатку у якому зберігаються дані об’єктів програми у випадку вибору користувачем команди «зберегти дані», яку реалізує метод Save.

Лістинг 3. – Реалізація поля fileName та методу Save класу DataContext

public static string fileName = "MacroregionsData.xml";

public static void Save() {

using (FileStream fSteam = File.Create(fileName)) {

XmlSerializer xmlSerializer =

new XmlSerializer(typeof(List<Country>));

xmlSerializer.Serialize(fSteam, objectsList);

}

}

Серіалізація відбувається шляхом виклику методу Serialize об’єкта xmlSerializer типу XmlSerializer, аргументконструктора якого визначає точний тип об’єкта, що серіалізується (наприклад, typeof(List<Country>).

Дані серіалізуються і відправляються у файловий потік fSteam, керування ресурсами якого здійснюється шляхом використання оператора using.

Завантаження даних з файлу відбувається автоматично на початку роботи програми у випадку, якщо файл з даними існує (лістинг 1. ).

Лістинг 1. – Реалізація поля методу Load класу DataContext

public static void Load() {

if (!File.Exists(fileName)) return;

using (FileStream fSteam = File.OpenRead(fileName)) {

XmlSerializer xmlSerializer =

new XmlSerializer(typeof(List<Country>));

var collection =

(IEnumerable<Country>)xmlSerializer

.Deserialize(fSteam);

objectsList.Clear();

objectsList.AddRange(collection);

}

}

Щоб зробити об'єкт доступним сервісу серіалізації .NET, досить помітити кожний зв'язаний з ним клас атрибутом [Serializable].

Якщо деякі члени даного класу не повинні (чи не можуть) брати участь у процесі серіалізації, то потрібно позначити відповідні поля атрибутом [NonSerialized]. Це може бути корисно тоді, коли в класі, призначеному для серіалізації, є члени-змінні, які запам'ятовувати не потрібно.

Щоб значення поля об’єкта не брало участь у процесі Xml-серіалізації, потрібно застосувати до відповідної властивості атрибут [XmlIgnore]:

XmlSerializer серіалізує та десеріалізує клас тільки у випадку, якщо для нього визначено конструктор, що використовується за замовчуванням.

## Реалізація меню та циклу виконання програми

Для реалізації

## ….

Для реалізації….

# Висновки

Створене проєктне рішення реалізовано мовою програмування С# в інтегрованому середовищі розробки Microsoft Visual Studio.

# Список використаних джерел

1. Коноваленко І. В. Програмування мовою C# 6.0. Навчальний посібник для технічних спеціальностей вищих навчальних закладів. Тернопіль. ТНТУ. 2016. 227 с.
2. UN data. URL: http://data.un.org/ (дата звернення 11.08.2023).
3. Чисельність населення на 1 січня 2014 року та середня чисельність за 2013 рік. Державна служба статистики України. URL: http://ukrstat.gov.ua/operativ/operativ2014/ds/kn/kn\_u/kn0114\_u.html (дата звернення 21.09.2023).
4. Чисельність населення України скоротилася до 42,9 млн осіб – Держстат. URL: http://tyzhden.ua/News/132248 (дата звернення 21.09.2023).
5. List of countries and dependencies by population. URL: https://en.wikipedia.org/wiki/List\_of\_countries\_and\_dependencies\_by\_population (дата звернення 03.10.2023).
6. Національна статистична класифікація «Класифікація країн Світу». URL: http://document.ua/pro-zatverdzhennja-klasifikaciyi-krayin-svitu-doc174792.html (дата звернення 11.10.2023).
7. Україна. URL: https://uk.wikipedia.org/wiki/Україна (дата звернення 11.10.2023).
8. Список країн за площею. URL: https://uk.wikipedia.org/wiki/Список\_країн\_за\_площею (дата звернення 11.10.2023).
9. Світ. URL: https://uk.wikipedia.org/wiki/Світ (дата звернення 14.10.2023).
10. Азія. URL: https://uk.wikipedia.org/wiki/Азія (дата звернення 14.10.2023).
11. Європа. URL: https://uk.wikipedia.org/wiki/Європа (дата звернення 14.10.2023).
12. Адміністративний поділ країн Європи. URL: https://uk.wikipedia.org/wiki/Адміністративний\_поділ\_країн\_Європи (дата звернення 14.10.2023).
13. UN Data. A World of information. URL: https://data.un.org/ (дата звернення 21.09.2023).

# Додатки

## Додаток А

Лістинг А – Опис класу Country

using Common.Entities;

using Common.Extensions;

namespace CountriesInfo {

[Serializable]

public class Country : Entity {

public string? Name { get; set; }

public Macroregion? Macroregion { get; set; }

public string? Alpha2 { get; set; }

public double? Area { get; set; }

public string? Note { get; set; }

public string? Description { get; set; }

public Country(string? name, Macroregion? macroregion,

string? alpha2, double? area) {

Name = name;

Macroregion = macroregion;

Alpha2 = alpha2;

Area = area;

}

public Country(string name, Macroregion macroregion)

: this(name, macroregion, null, null) { }

public Country() : this(null, null, null, null) { }

public override string MembersString =>

$"{MembersIndent}Назва: {Name}\n" +

$"{MembersIndent}Географічний регіон: {Macroregion?.Name}\n" +

$"{MembersIndent}Код alpha-2 ISO: {Alpha2}\n" +

$"{MembersIndent}Площа, км: {Area}\n" +

$"{MembersIndent}Примітка: {Note?.ToTextBlocks(ValuesIndent, "\n")}\n" +

$"{MembersIndent}Опис: {Description?.ToTextBlocks(ValuesIndent, "\n")}";

public override string Key => Name ?? base.Key;//

}

}

## Додаток Б

Лістинг Б – Опис класу Macroregion

using Common.Entities;

using Common.Entities.Interfaces;

using Common.Extensions;

namespace CountriesInfo {

[Serializable]

public class Macroregion : Entity, IHierarchical<Macroregion> {

public string? Name { get; set; }

public Macroregion? Parent { get; set; }

public double? Area { get; set; }

public string? Description { get; set; }

public override string MembersString =>

$"{MembersIndent}Назва: {Name}\n" +

$"{MembersIndent}Макрорегіон: {Parent?.Name}\n" +

$"{MembersIndent}Площа, км: {Area}\n" +

$"{MembersIndent}Опис: {Description?.ToTextBlocks(ValuesIndent, "\n")}";

public Macroregion(string? name, Macroregion? parent, double? area) {

Name = name;

Parent = parent;

Area = area;

}

public Macroregion() : this(null, null, null) { }

public Macroregion(string name, Macroregion? parent)

: this(name, parent, null) { }

public override string Key => Name ?? base.Key; //

}

}

## Додаток В

Лістинг В – Реалізація команд зчитування та запису даних з файлу

using CountriesInfo.Data.Interfaces;

using System.Text;

using System.Xml;

namespace CountriesInfo.Data.IO {

public class XmlFileIoController : IFileIoController {

public string FileTypeCaption => "Файли формату XNL";

public string FileExtension => ".xml";

public void Save(IReadOnlyDataSet dataSet,string filePath){

Path.ChangeExtension(filePath, FileExtension);

XmlWriterSettings settings = new XmlWriterSettings();

settings.Encoding = Encoding.Unicode;

XmlWriter? writer = null;

try {

writer = XmlWriter.Create(filePath, settings);

WriteData(dataSet, writer);

}

catch (Exception) {

throw;

}

finally {

if (writer != null)

writer.Close();

}

}

void WriteData(IReadOnlyDataSet dataSet, XmlWriter writer){

writer.WriteStartDocument();

writer.WriteStartElement("CountriesInfo");

WriteMacroregions(dataSet.Macroregions, writer);

WriteCountries(dataSet.Countries, writer);

writer.WriteEndElement();

writer.WriteEndDocument();

}

void WriteMacroregions(IEnumerable<Macroregion> collection, XmlWriter writer) {

writer.WriteStartElement("MacroregionsData");

foreach (var inst in collection) {

writer.WriteStartElement("Macroregion");

writer.WriteElementString("Id", inst.Id.ToString());

writer.WriteElementString("Name", inst.Name);

int parentId = inst.Parent == null ? 0 : inst.Parent.Id;

writer.WriteElementString("ParentId", parentId.ToString());

writer.WriteElementString("Area", inst.Area.ToString());

writer.WriteElementString("Description", inst.Description);

writer.WriteEndElement();

}

writer.WriteEndElement();

}

private void WriteCountries(IEnumerable<Country> collection, XmlWriter writer) {

writer.WriteStartElement("CountriesData");

foreach (var inst in collection) {

writer.WriteStartElement("Country");

writer.WriteElementString("Id", inst.Id.ToString());

writer.WriteElementString("Name", inst.Name);

int macroregionId = inst.Macroregion == null ? 0 : inst.Macroregion.Id;

writer.WriteElementString("MacroregionId", macroregionId.ToString());

writer.WriteElementString("Alpha2", inst.Alpha2);

writer.WriteElementString("Area", inst.Area.ToString());

writer.WriteElementString("Note", inst.Note);

writer.WriteElementString("Description", inst.Description);

writer.WriteEndElement();

}

writer.WriteEndElement();

}

//---------------------------------------------------------

public bool Load(IDataSet dataSet, string filePath) {

Path.ChangeExtension(filePath, FileExtension);

if (!File.Exists(filePath)) return false;

XmlReaderSettings settings = new XmlReaderSettings();

settings.IgnoreWhitespace = true;

using (XmlReader reader = XmlReader.Create(filePath, settings)) {

while (reader.Read()) {

if (reader.NodeType == XmlNodeType.Element) {

switch (reader.Name) {

case "Macroregion":

ReadMacroregion(reader, dataSet.Macroregions);

break;

case "Country":

ReadCountry(reader, dataSet);

break;

}

}

}

}

return true;

}

void ReadMacroregion(XmlReader reader, ICollection<Macroregion> collection) {

reader.ReadStartElement("Macroregion");

int id = reader.ReadElementContentAsInt();

string name = reader.ReadElementContentAsString();

int parentId = reader.ReadElementContentAsInt();

Macroregion? parent = collection.FirstOrDefault(e => e.Id == parentId);

Macroregion inst = new Macroregion(name, parent){Id = id };

string s = reader.ReadElementContentAsString();

inst.Area = string.IsNullOrEmpty(s) ? null : double.Parse(s);

inst.Description = reader.ReadElementContentAsString();

collection.Add(inst);

}

void ReadCountry(XmlReader reader, IDataSet dataSet) {

Country inst = new Country();

reader.ReadStartElement("Country");

inst.Id = reader.ReadElementContentAsInt();

inst.Name = reader.ReadElementContentAsString();

int macroregionId = reader.ReadElementContentAsInt();

inst.Macroregion = dataSet.Macroregions

.FirstOrDefault(e => e.Id == macroregionId);

inst.Alpha2 = reader.ReadElementContentAsString();

string s = reader.ReadElementContentAsString();

inst.Area = string.IsNullOrEmpty(s) ? null : double.Parse(s);

inst.Note = reader.ReadElementContentAsString();

inst.Description = reader.ReadElementContentAsString();

dataSet.Countries.Add(inst);

}

}

}

## Додаток Г

Лістинг Г – Реалізація команд сортування даних

using System;

using System.Linq;

namespace CountriesInfo{

partial class Editor{

private void OutData() {

OutMacroregionsData();

OutCountriesData();

}

private void OutMacroregionsData() {

Console.WriteLine(" Список географічних регіонів:");

foreach (var obj in sortingMacroregions) {

Console.WriteLine("{0,5} {1,-17}", obj.Id, obj.name);

}

}

private void OutCountriesData() {

Console.WriteLine("\n Список країн:");

foreach (var obj in sortingCountries) {

string macroregionName = obj.macroregion == null ? ""

: obj.macroregion.name;

Console.WriteLine("{0,5} {1,-24} {2,-17}, код: {3,2}, площа: {4,8} км²",

obj.Id, obj.name, macroregionName, obj.alpha2, obj.area);

}

}

public void AddMacroregion() {

Macroregion inst = new Macroregion();

inst.name = Entering.EnterString("Назва географічного регіону");

inst.Id = dataContext.Macroregions.Select(e => e.Id).Max() + 1;

dataContext.Macroregions.Add(inst);

}

public void AddCountry() {

Country inst = new Country();

inst.name = Entering.EnterString("Назва країни");

inst.macroregion = SelectMacroregion();

inst.alpha2 = Entering.EnterStringOrNull("Код alpha-2");

inst.area = Entering.EnterNullableDouble("Площа");

inst.Id = dataContext.Countries.Select(e => e.Id).Max() + 1;

dataContext.Countries.Add(inst);

}

private Macroregion SelectMacroregion() {

string macroregionName = Entering.EnterString("Географічний регіон");

return dataContext.Macroregions.First(e => e.name == macroregionName);

}

public void RemoveMacroregion() {

int id = Entering.EnterInt32("Введіть числовий ідентифікатор регіону");

Macroregion inst = dataContext.Macroregions.FirstOrDefault(e => e.Id == id);

dataContext.Macroregions.Remove(inst);

}

public void RemoveCountry() {

int id = Entering.EnterInt32("Введіть числовий ідентифікатор країни");

Country inst = dataContext.Countries.FirstOrDefault(e => e.Id == id);

dataContext.Countries.Remove(inst);

}

private void SortMacroregionsByName() {

sortingMacroregions = sortingMacroregions.OrderBy(e => e.name);

}

private void SortCountriesByName() {

sortingCountries = sortingCountries.OrderBy(e => e.name);

}

private void SortCountriesByMacroregion() {

sortingCountries = sortingCountries.OrderBy(e => e.macroregion.name);

}

private void SortCountriesByAlpha2() {

sortingCountries = sortingCountries.OrderBy(e => e.alpha2);

}

private void SortCountriesByArea() {

sortingCountries = sortingCountries.OrderBy(e => e.area);

}

}

}

## Додаток Д

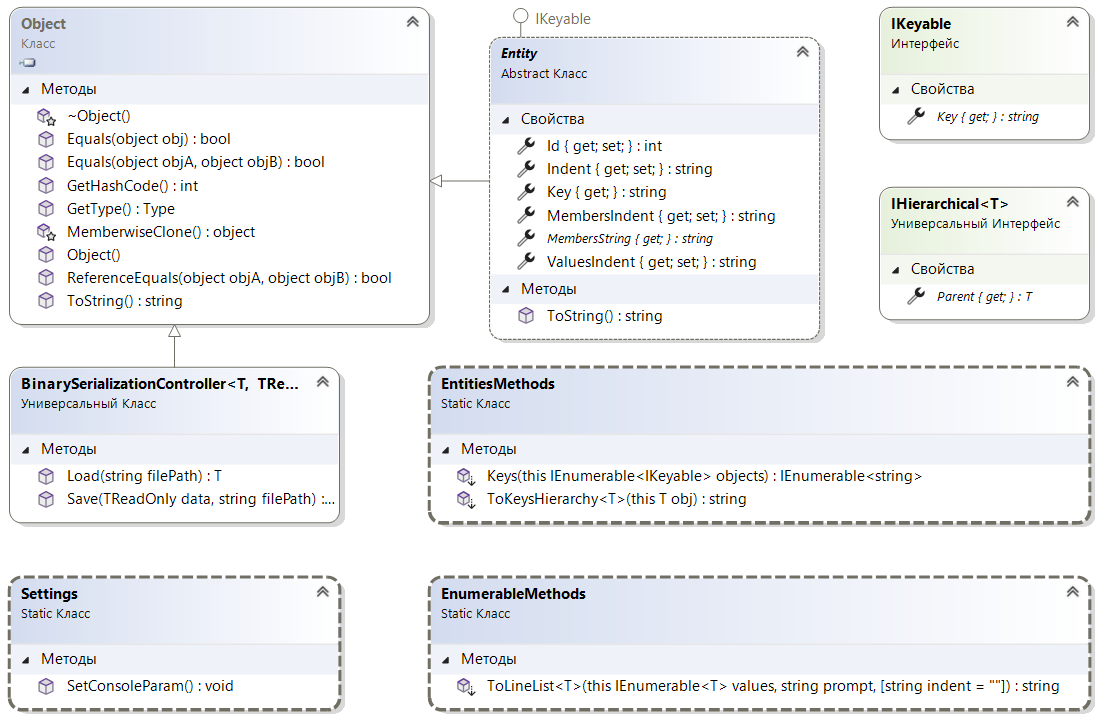


Рисунок Д – Діаграма класів …

## Додаток Е

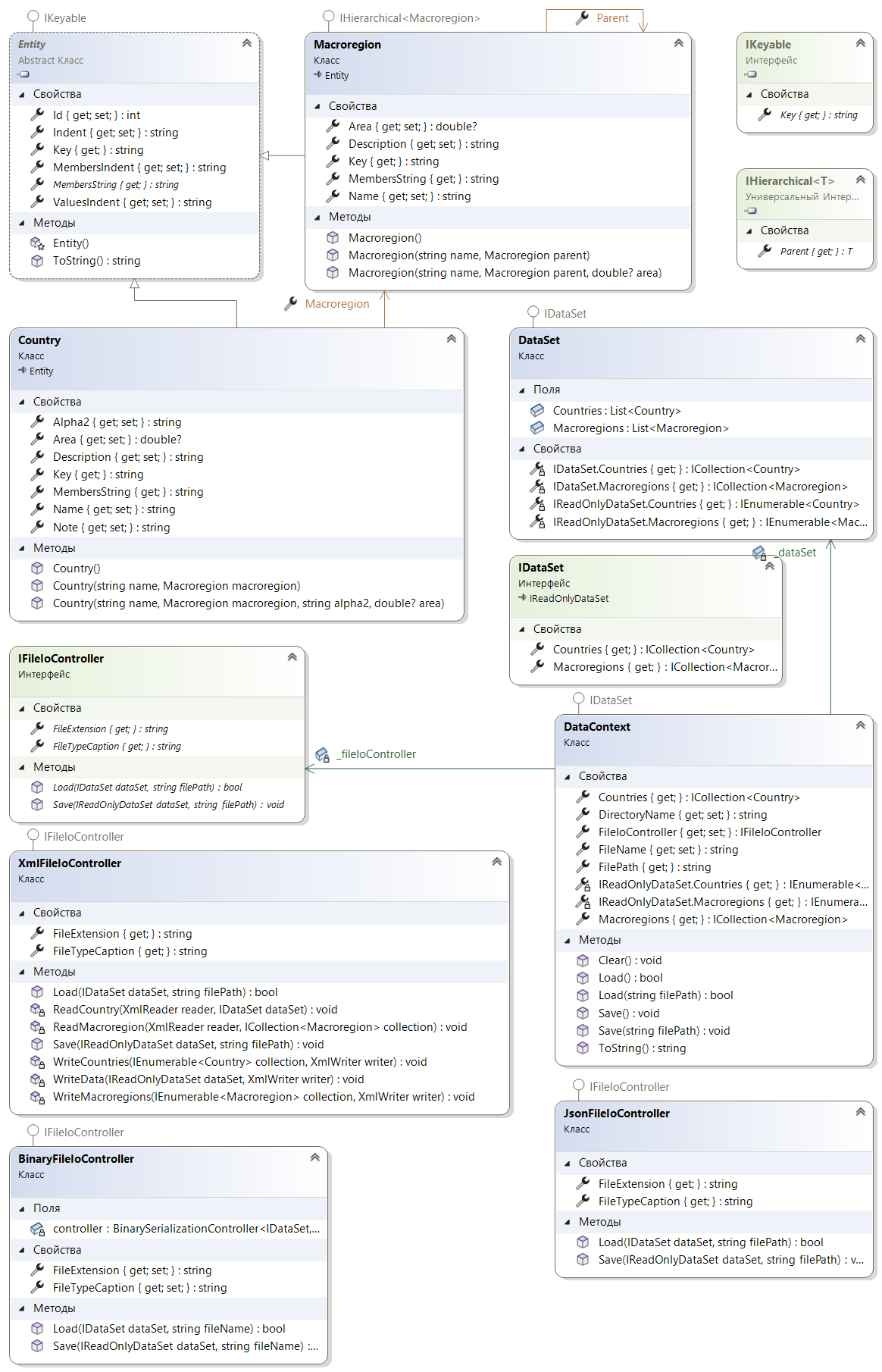


Рисунок Е – Діаграма класів …

## Додаток Ж

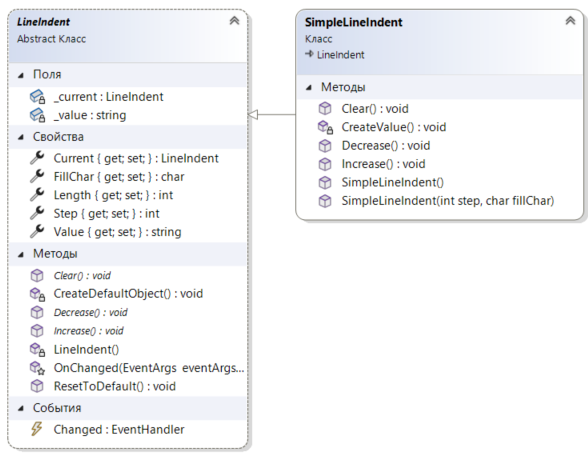


Рисунок Ж – Діаграма класів …